

By the end of this lesson students will be able to:

- Create a mathematical model of a single neuron (perceptron).
- Calculate output from a single neuron.
- Given a neuron, determine the equation for a line that separates two classes of inputs.
- Distinguish between linearly and non-linearly separable data.
- Describe how a neural network can classify non-linearly separable data.

I. Overview of bioinformatics

Bioinformatics

Coursework

- Comp Sci
- Statistics
- Math

BME Area

- Bioinformatics

Industry

Established industry: Yes

Areas

- Health informatics
- Image processing
- Biomedical research

Research

- -omics
- Computational biology
- Healthcare informatics

Bioinformatics is...

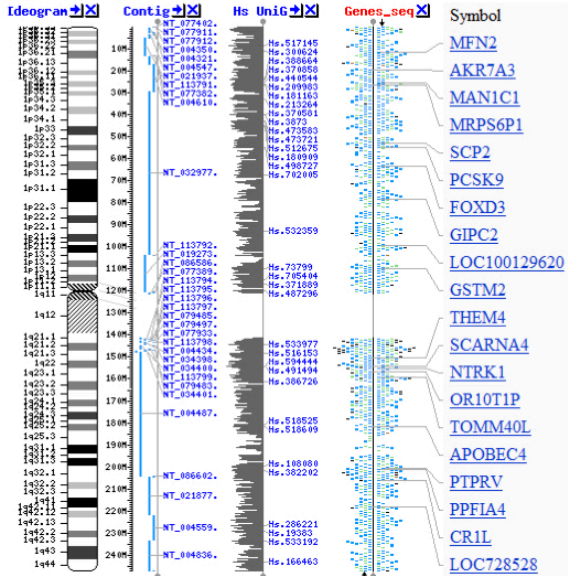
- Informatics
 - Methods for analyzing data
- Bioinformatics
 - Methods for analyzing biological data
- Computational biology
 - Use computer algorithms/modeling solve biological problems
- Systems biology
 - Study of molecular and cellular systems

[Homo sapiens \(human\) Build 36.3 \(Current\)](#)

Chromosome: [1] 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 X

Master Map: Genes On Sequence [Sum](#)

Region Displayed: 0-247M bp



Are these sequences related?
How do we align them?

GACGGATTAG
GATCGGAATAG

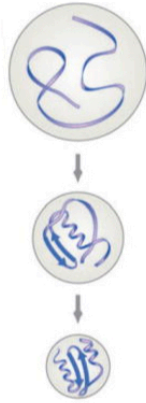
Algorithms to compare DNA
Sequences

GA-CGGATTAG
GATCGGAATAG

Protein folding problem:

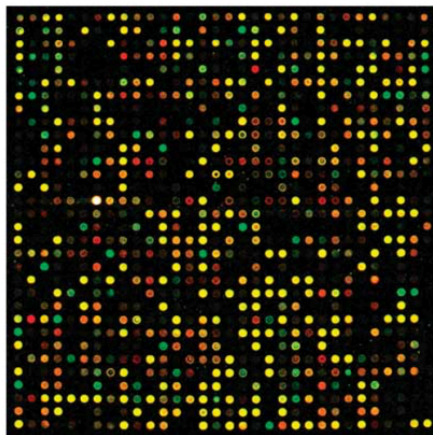
Can we predict the 3D shape of a protein from its AA sequence?

GIVEQCCTSICSLYQLENYCN...



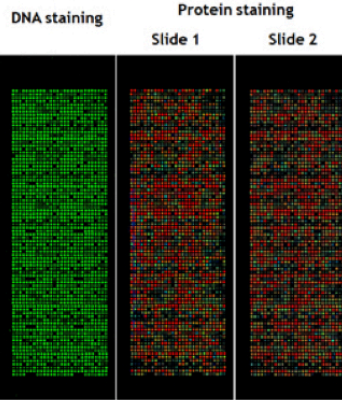
High throughput experiments

genomics, proteomics, metabolomics



DNA microarray

Protein Display using Human Lysate



DNA microarray

<http://nappaproteinarray.org/technology.html>

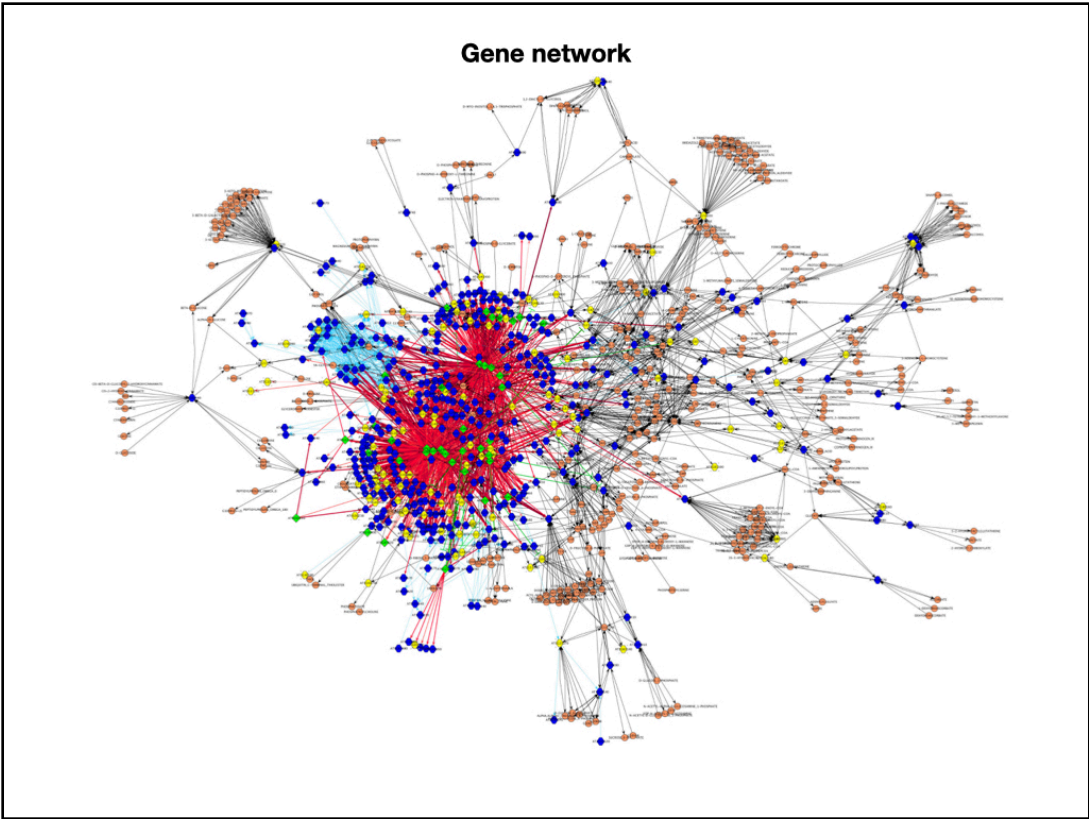
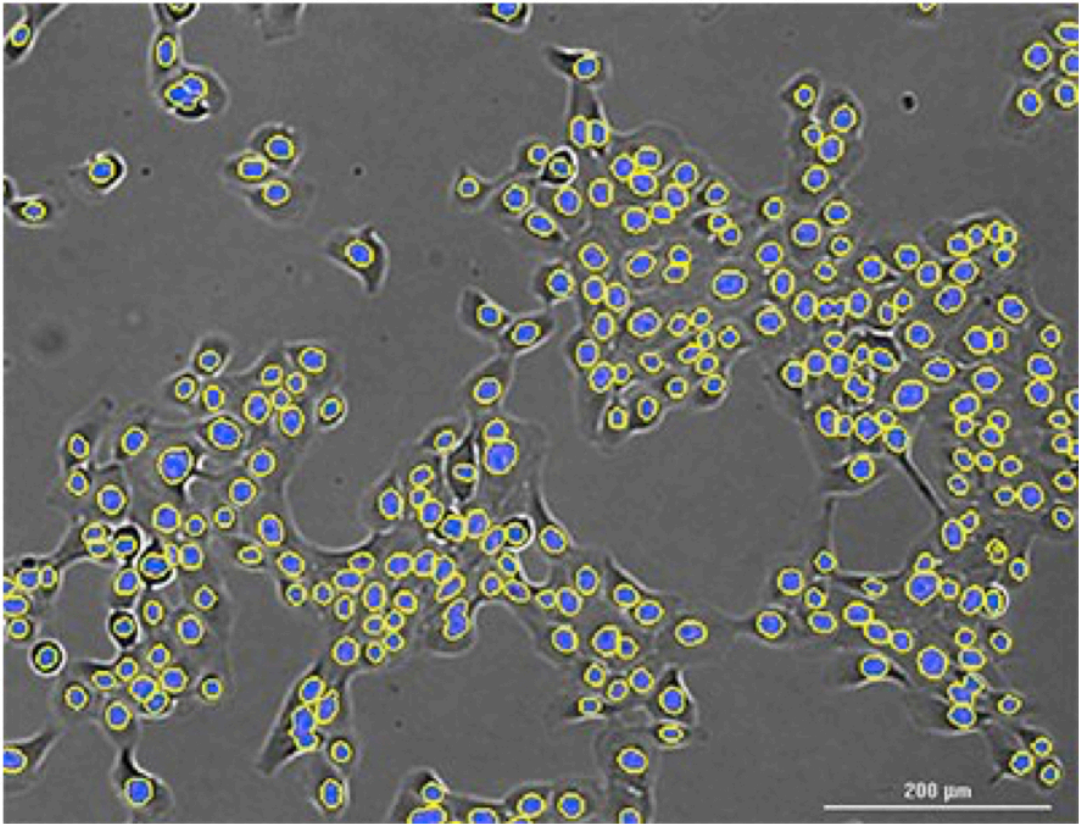


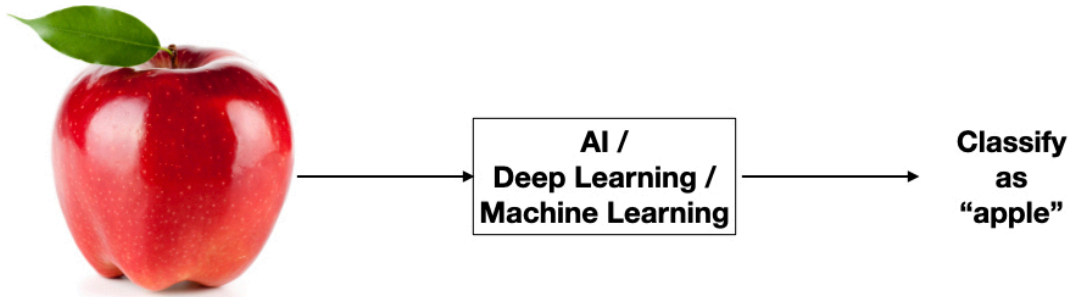
Image analysis



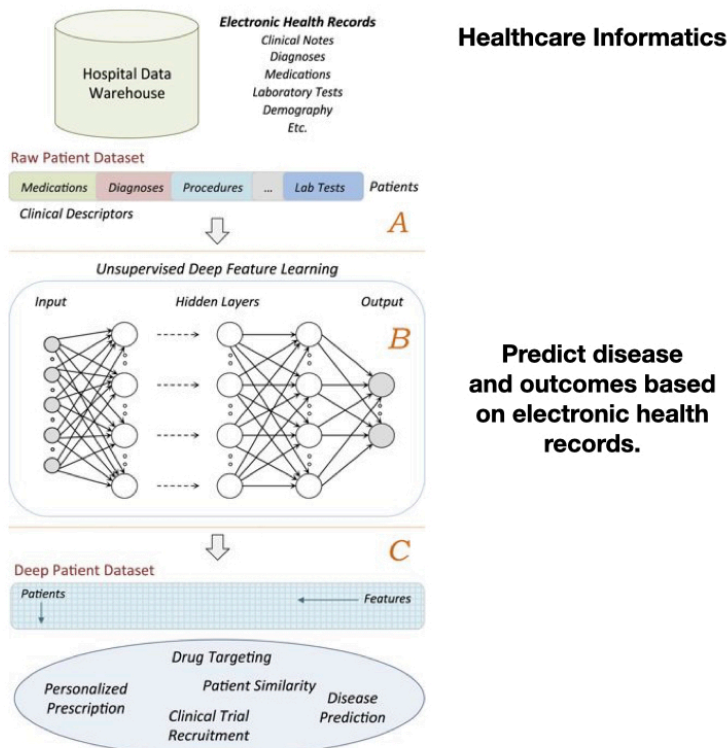
<https://www.biotek.com/applications/cell-counting.html>

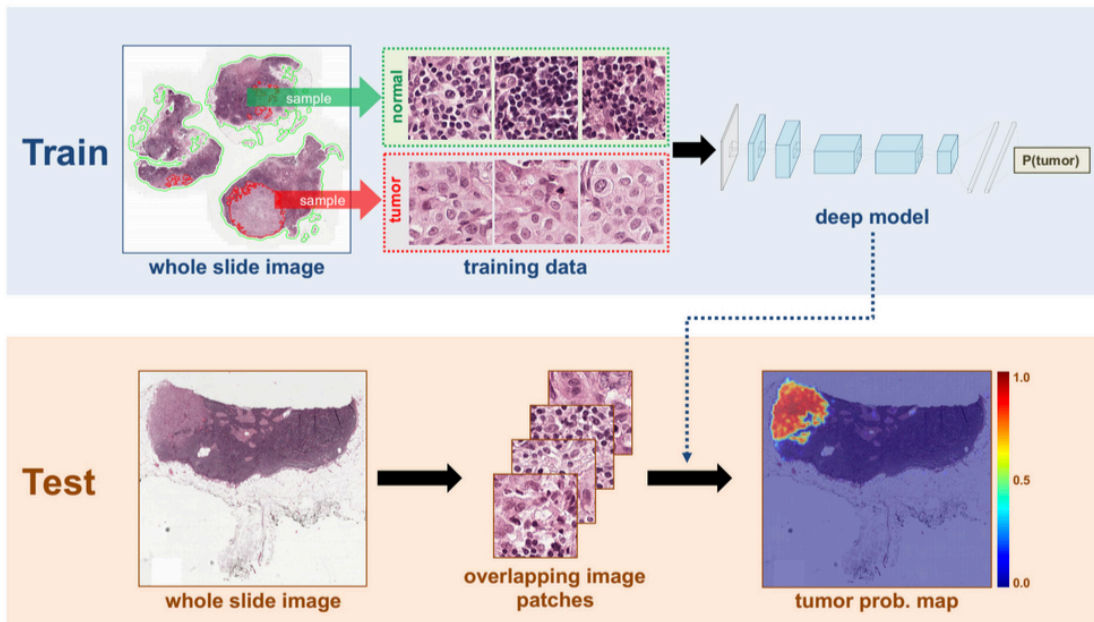
II. Examples of AI/Deep learning/machine learning applied to BME

- A. Many examples of AI/machine learning are really complex and creative applications of pattern recognition
- B. When you hear these terms, don't think of self-aware programs and robots
- C. Rather, think of a computer that is really good at matching something that it has never seen before to something that it has
- D. For example



- E. What do we use as inputs?
 - 1. Maybe an image of the apple. The AI would then analyze each pixel of the image and match it to known patterns that are also apples.
 - 2. Maybe some characteristics of the apple input by a user. For example the words and phrases "red" "fruit" "grows on tree". The AI would then match these inputs to known patterns that are apples.
- F. But what's inside the box? What does AI/Deep Learning/Machine Learning mean?
 - 1. Generally, any pattern matching algorithm
 - 2. A common one is known as an artificial neural network (ANN)
- G. Here are some examples of ANNs applied to BME problems:

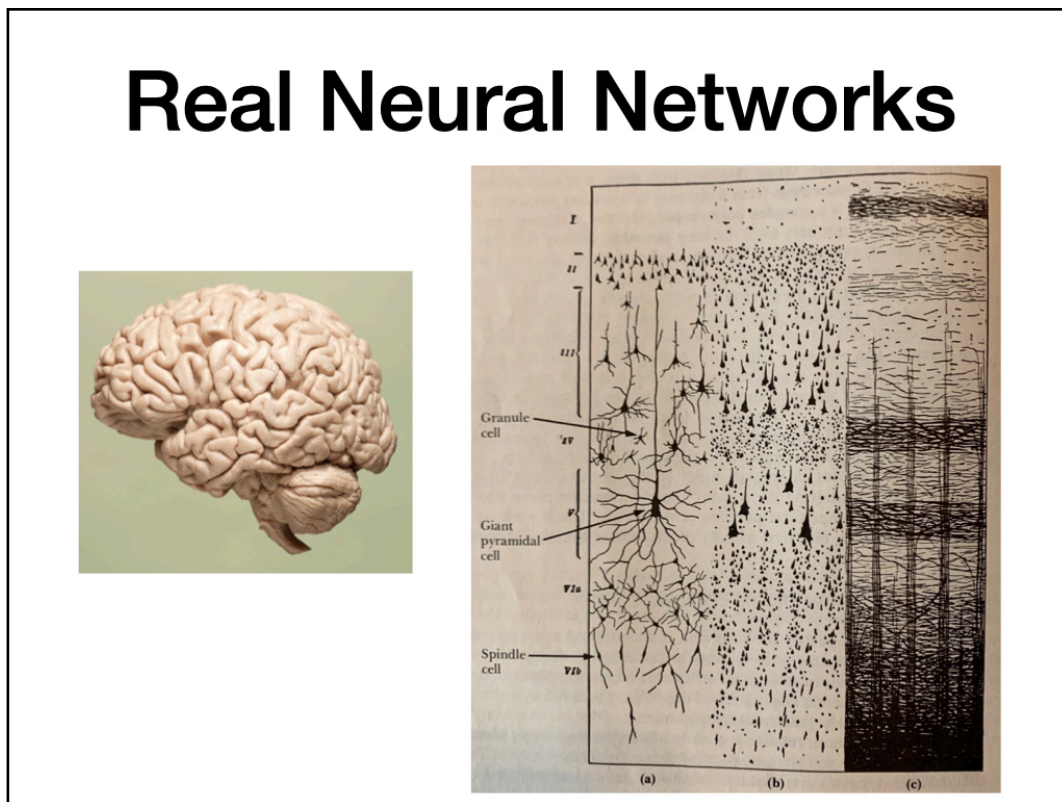


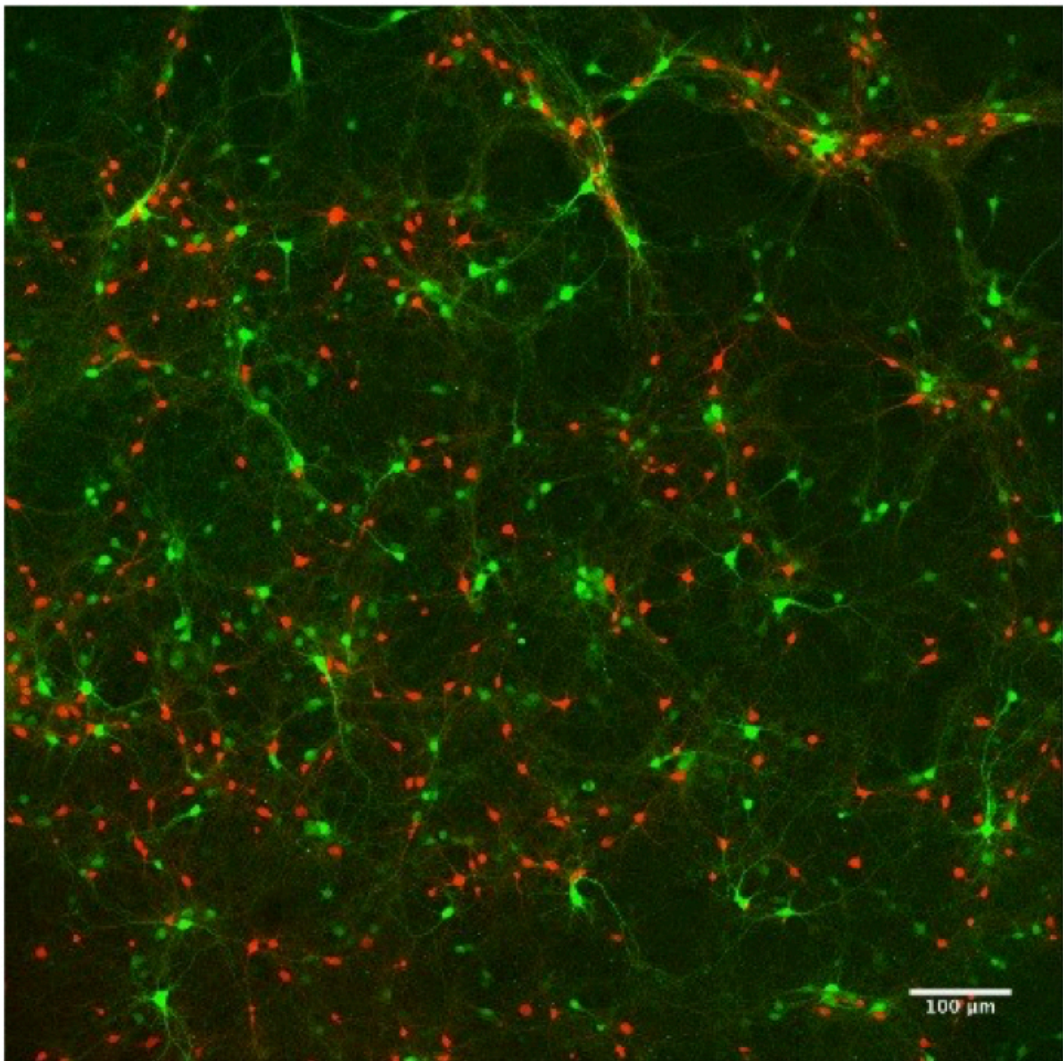
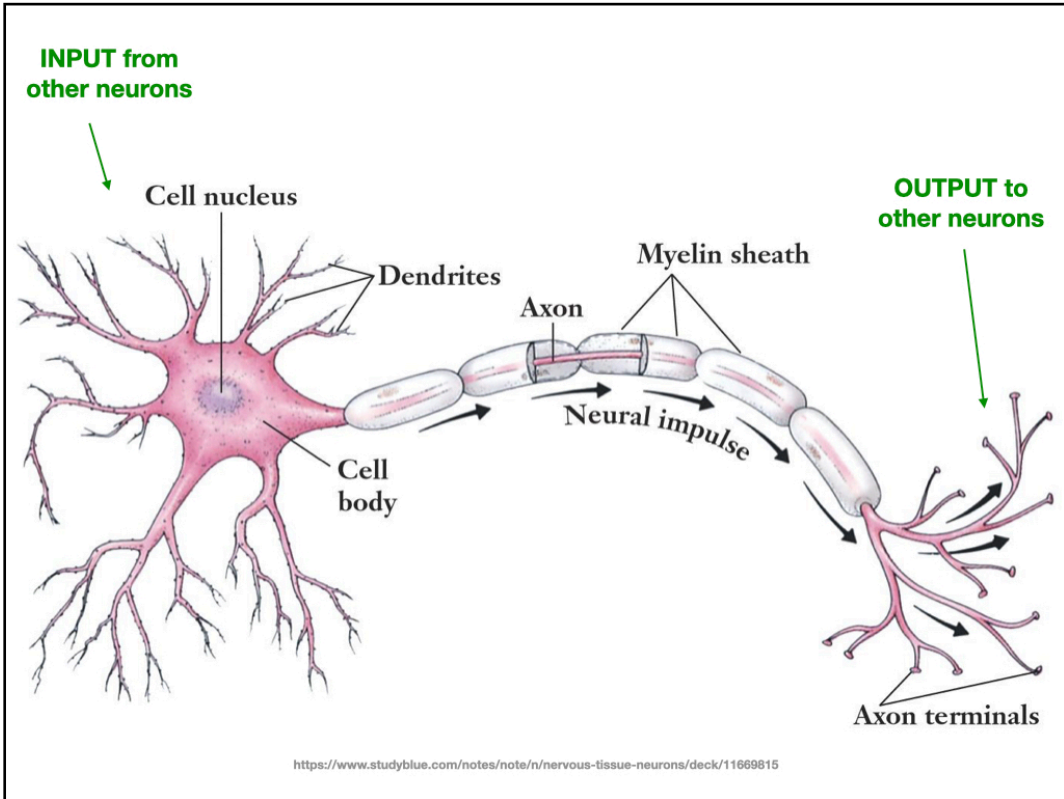


<https://news.developer.nvidia.com/wp-content/uploads/2016/06/DL-Breast-Cancer-Detection-Image.png>

III. Neural networks (also called Artificial Neural Networks, or ANNs)

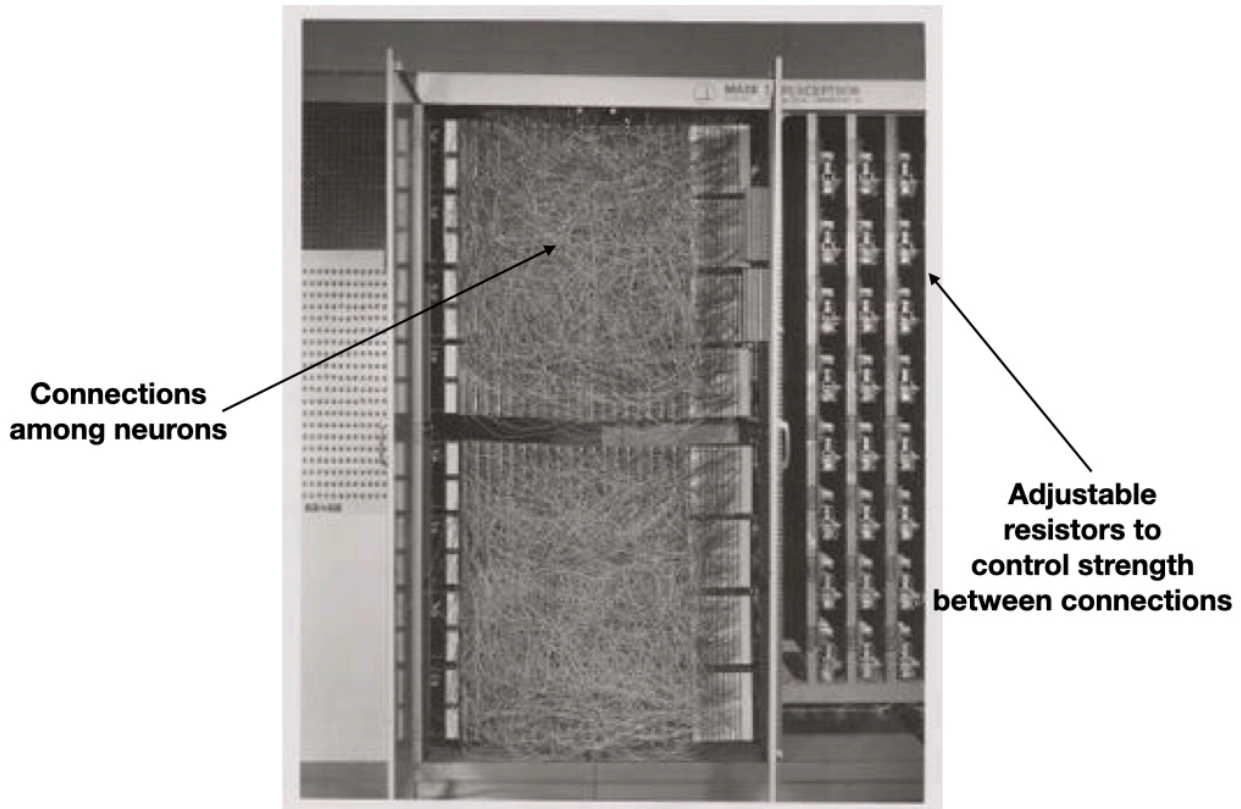
A. Computers or computer programs that mimic the behavior of neuron networks





B. Original ANN was built out of hardware and was used to recognize images

**First Artificial Neural Network
400 photocells for inputs**



D. Now, ANNs are available as software in Matlab, Python, etc.

E. The building block of a NN is the neuron

F. The neuron works like this:

1. Dendrites are hooked to other neurons that either provide a pulse or do not
2. The neuron takes all these inputs and adds them together
3. If there are enough pulses, then the neuron fires
4. This neuron is also hooked to other neurons downstream, and so on

G. I can create a mathematical model of the neuron like this:

1. The function can be lots of things, but we choose this one for now:

2. The function can also be non-linear. A common function is the sigmoid and if you use this function the model is called the Pitts McCullough model.

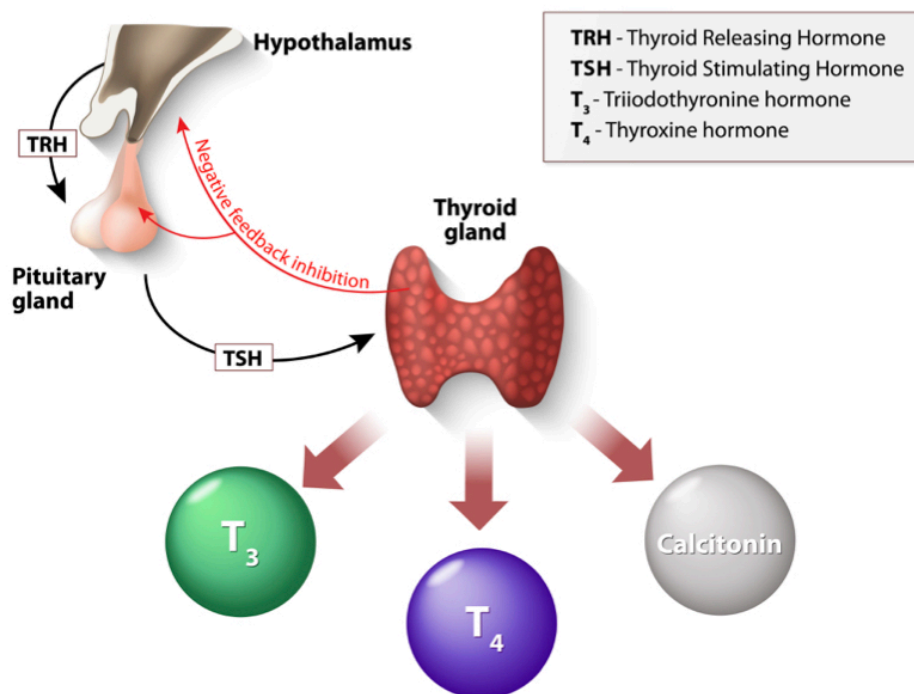
- H. The simplest type of neural network is just a single neuron. Even though it's simple, it is still very useful and can classify things.
- I. Here's how it works

- J. This type of NN is called a *single-layer perceptron* and it's good enough to classify two or more inputs into one or more groups
- K. Students example. Consider a single-layer perceptron with $w_1 = w_2 = 1$ and $\Theta = 0.5$ and only two inputs, x_1 and x_2 . Calculate the output for the inputs $x = [0,0]$, $x = [0,1]$, $x = [1,0]$, $x = [1,1]$.

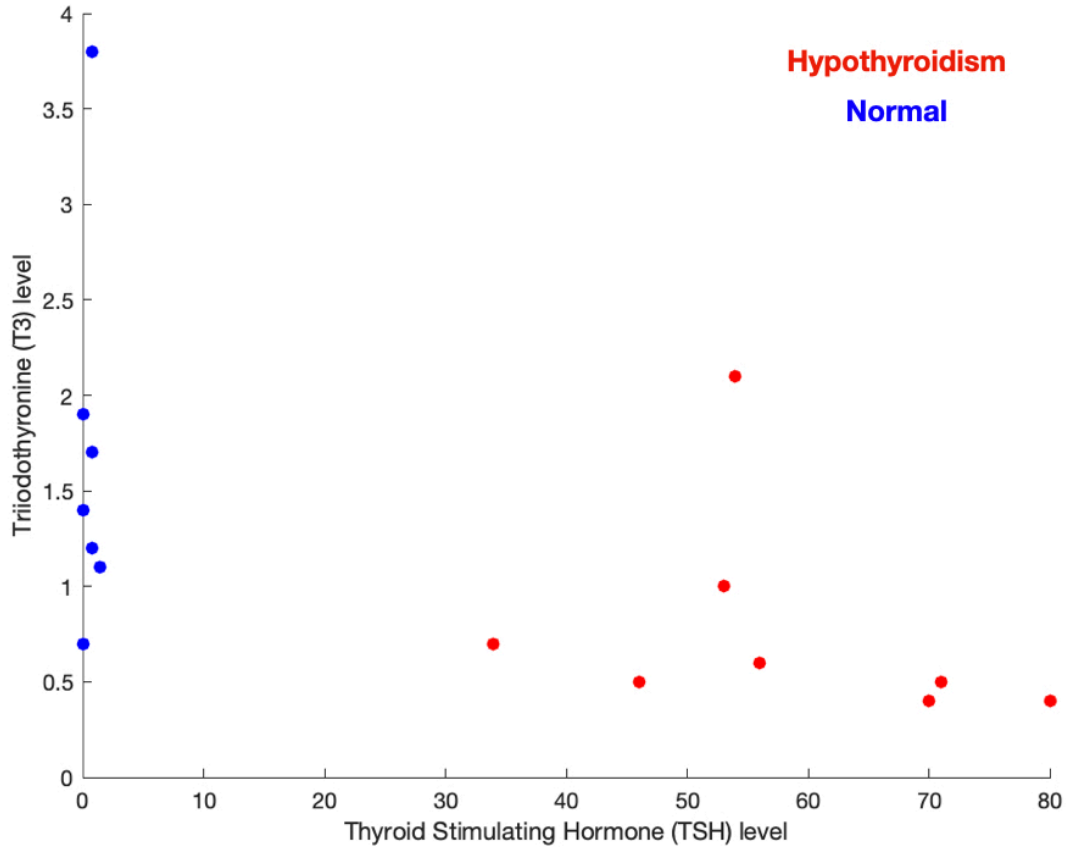
- L. Now, here are all of the inputs from the previous problem, plotted on the axes x_1 and x_2 . Look at the weights. If we solve the equation we can see that it makes a line that separates the two groups of dots

- M. Here is a real-life example using data for people with and without hypothyroidism.

THYROID HORMONES

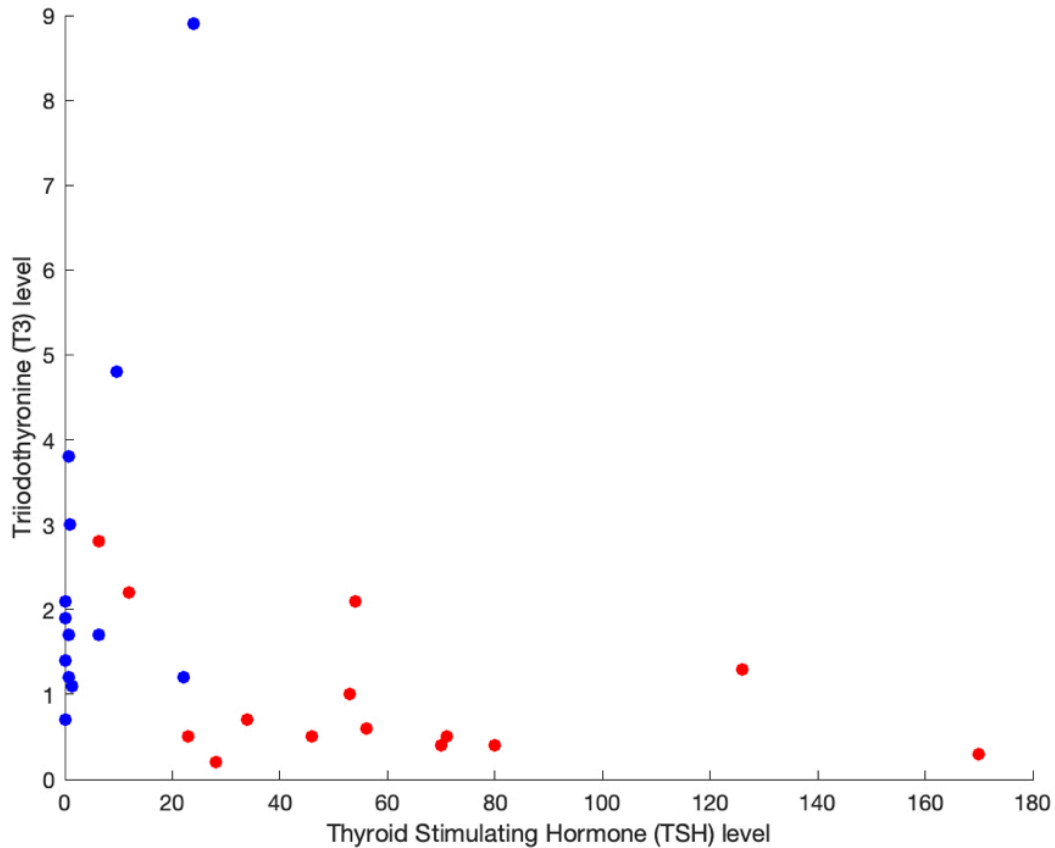


N. The object is to come up with a NN that will diagnose hypothyroidism. The inputs to the NN will be T_3 and TSH levels. Here is some real clinical data from 15 patients:

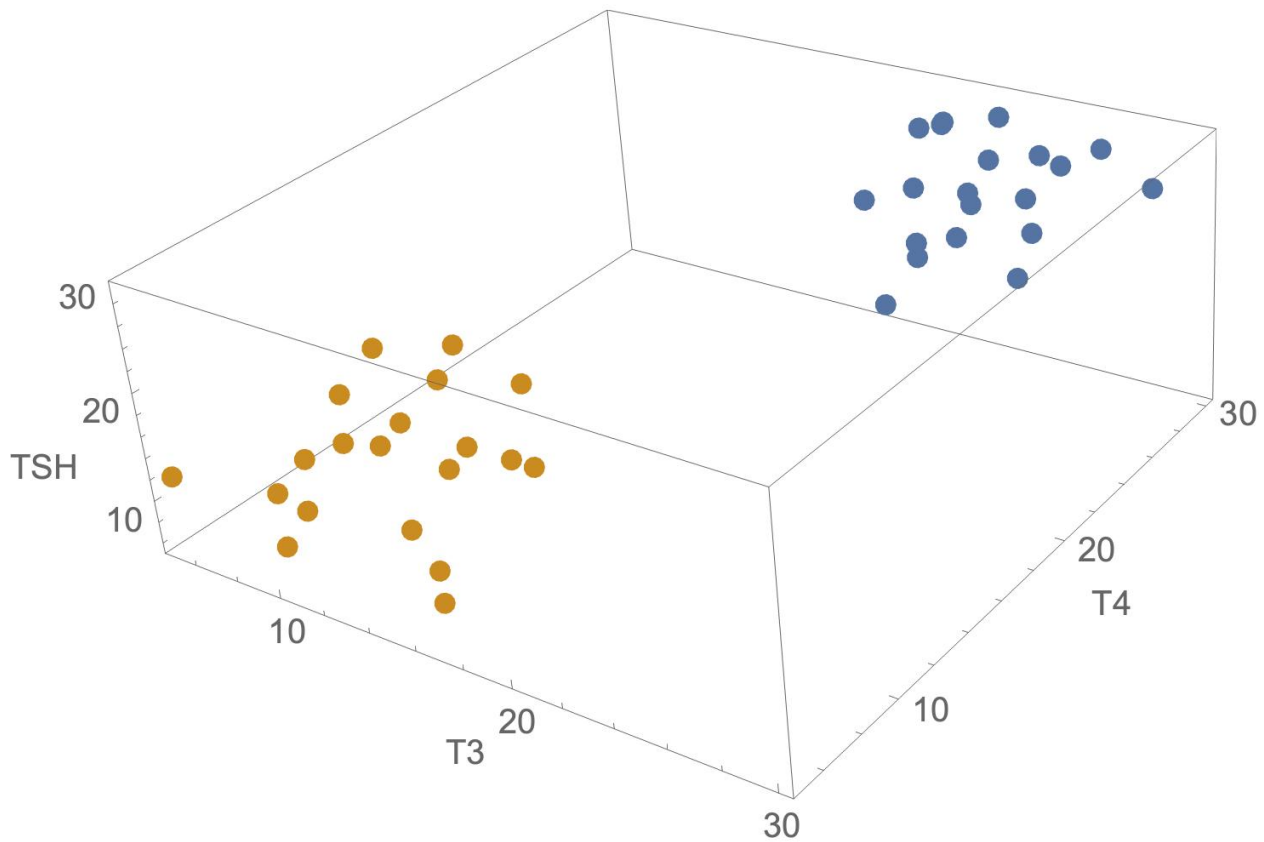


- O. How do we build a NN to do the classification?
- P. Student exercise. What are some possible lines that could be used to separate the two groups?
- Q. Student exercise. Pick a line that runs through the origin and falls on points (20,1) and (40,2). Come up with an equation for the line. Translate into an equation for a NN.

- R. This is all pretty easy so far, when the two groups are clearly distinct. Unfortunately in the real world data is almost never this easy to work with.
- S. Let's assume some more patients come into the clinic and we measure their hormone levels and add them to the graph. This is what it looks like for 27 patients:



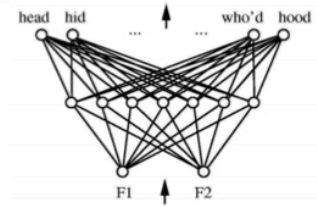
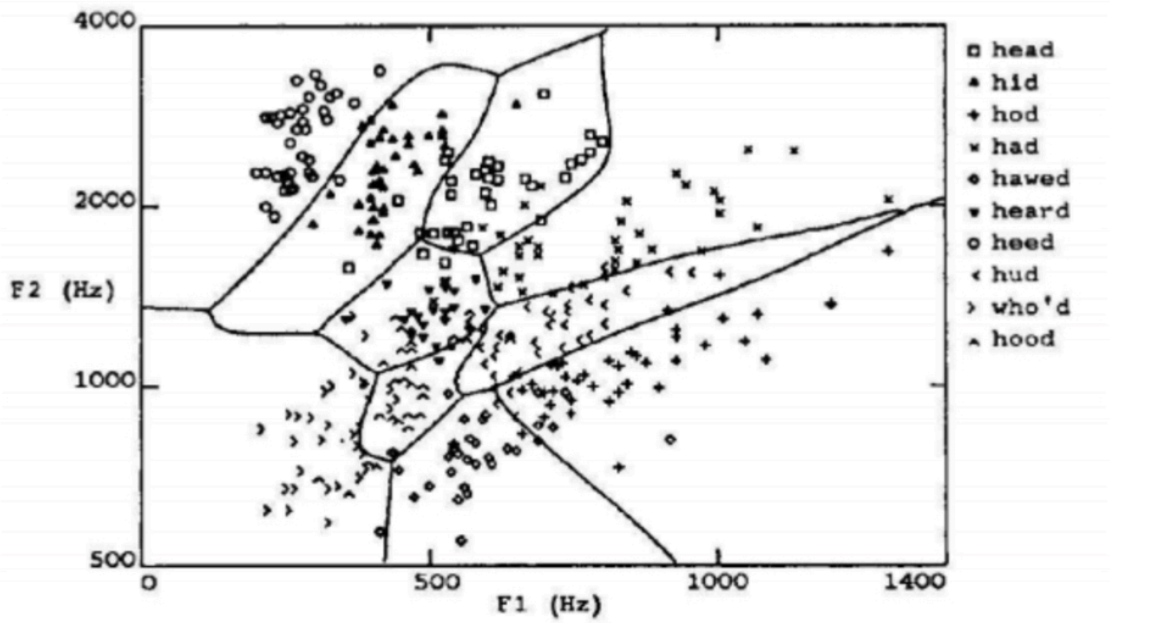
- T. Now where can I draw a line?
- U. There isn't such a clear distinction between the two groups any more.
 1. The data is no longer *linearly separable*, so a line will not separate the two groups with 100% accuracy
- V. I have three choices at this point
 1. Stick with a single line and know that my NN will have something less than 100% accuracy (i.e., it's wrong sometimes).
 2. Add another input, and/or
 3. Add a hidden layer of neurons
- W. Add another input
 1. What this does is makes my data 3 dimensional instead of 2 dimensional
 2. Assume that each input is not just T3 and TSH but T3, T4, and TSH. Then my plot might look like this:



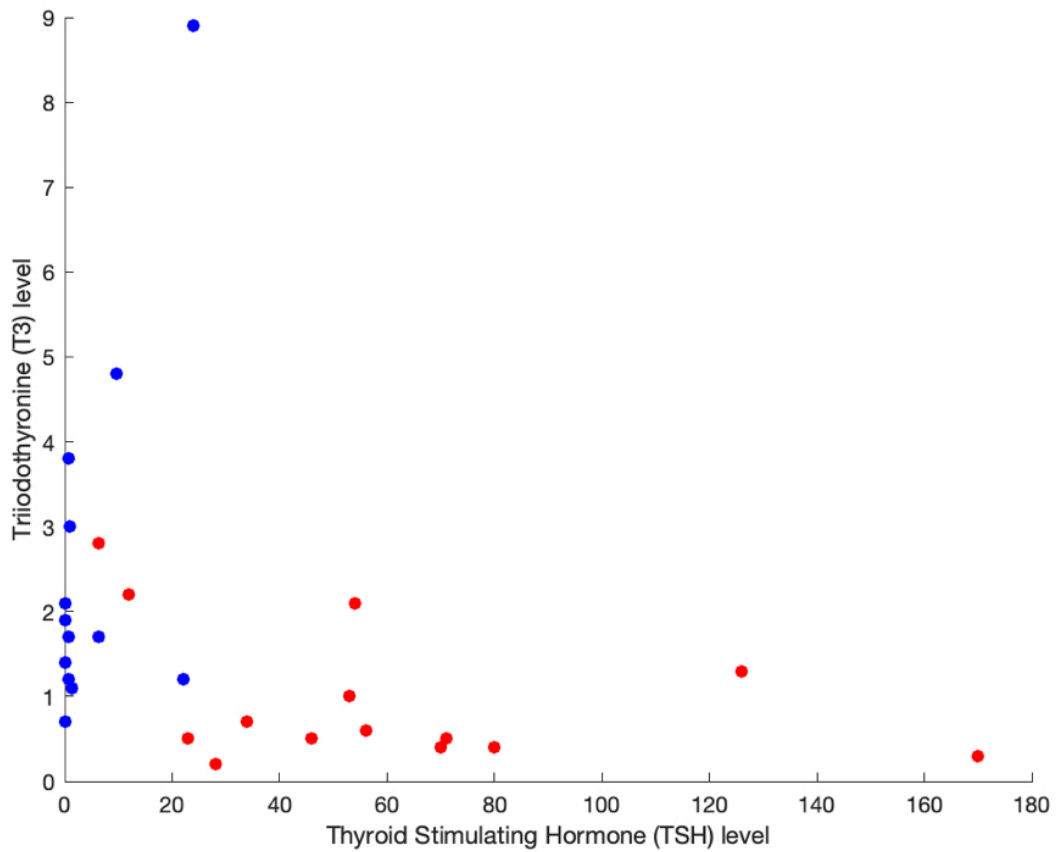
3. Now, I can see that there are “clouds” of data that can be separated by a plane
4. In summary:
 - a) 2d data is separated by a line
 - b) 3d data is separated by a plane
 - c) 4d data is separated by a hyperplane (a plane in $\geq 3d$)
5. Would now have an equation for these three inputs that looks like:

X. Hidden layers

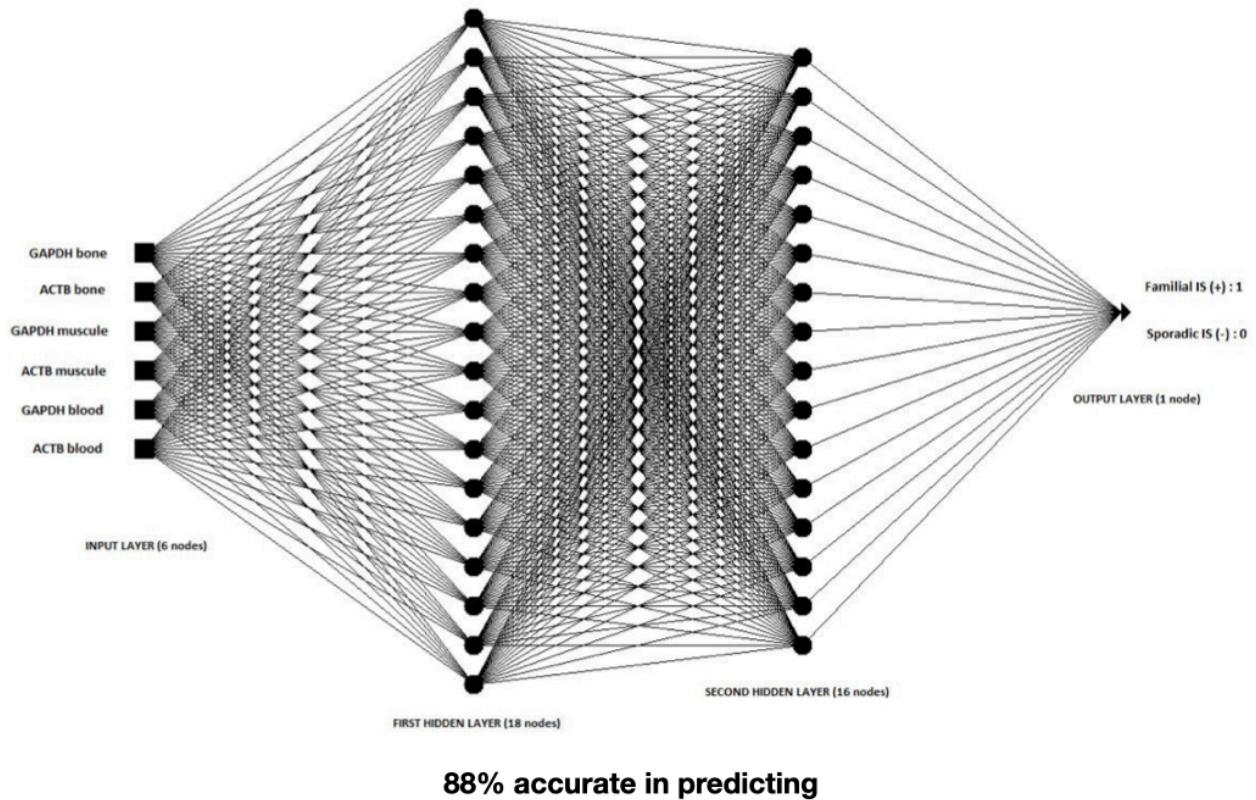
1. Going back a bit, the other option to separate the groups shown in letter S is to add hidden layers to the network. Hidden layers allow me to create series of lines segments of just one straight line or plane.
2. Here is an example of what adding hidden layers allows you to do:



3. With the T3 and TSH data, adding hidden layers we could do something like this:



4. Here is what the structure of a neural network with hidden layers looks like. This is called a *multilayer neural network* and this is what people usually are talking about when they say *deep learning*.



5. Almost all neural networks you hear about today are multilayer and have 10s-100s of nodes per layer (each neuron is called a node).

IV. How do you get the weights?

- A. We've seen two ways so far:
1. I gave you weights
 2. You calculated weights when it was a straight line
- B. In real-world problems, it's usually not practical to calculate the weights by hand, and if you have a multilayer neural network you cannot calculate them at all.
- C. You must use a *learning algorithm* where the neural network is given inputs with known outputs and adjusts the weights until its prediction accuracy is as close to 100% as possible.
1. This is known as *supervised learning* because you have given the neural network the inputs and outputs
- D. Here is a commonly used learning algorithm:
1. Initialize weights to random values
 2. For each input

- a) Calculate output
- b) Compare calculated output to correct output
- c) Adjust weights with
- d) Repeat a-c until the weights stop changing